80

Popular Computing

The world's only magazine devoted to the art of computing.

November 1979 Volume 7 Number 11

	Pound	French Franc	Deutsche Mark	Yen	Peso	Sol	Lira
Pound (2.1475)	...	.10836	.25145				
French Franc (.2327)	9.2286	...					
Deutsche Mark (.5400)	3.9769		...				
Yen (.004571)	469.81			...			
Peso (.0438)	49.030				...		
Sol (.004525)	474.59					...	
Lira (.001192)	801.59						...

Currency Exchange

Currency Exchange

PROBLEM 262

The chart on the cover shows, along the left-hand column, the value of seven currencies on June 21, 1979 in terms of U.S. dollars. For example, the Mexican Peso was worth $0.0438 that day.

The other columns of the chart show what one unit of the currency listed at the top will buy of each of the others. For example, one pound would buy 474 Peruvian Soles; one French Franc would buy .108 English Pounds, and so on.

Problem 1: Write a program whose input data is the set of figures in the left-hand column, and whose output is the complete body of the table.

Problem 2: Suppose that there is a sudden demand for Pesos in the United States, such that their price goes up to $.0472. Calculate the effect of that change (or any change of a similar nature) on all the other entries in the table.

Problem 3: Suppose the value of one currency changes suddenly, but the change is NOT reflected throughout the table. Calculate how one might make a profit by buying and selling currencies (neglecting any commissions on such sales for the moment).

Publisher: **Audrey Gruenberger**
Editor: **Fred Gruenberger**
Associate Editors: **David Babcock**
Irwin Greenwald
Patrick Hall
Contributing Editors: **Richard Andree**
William C. McGee
Thomas R. Parkin
Edward Ryan
Art Director: **John G. Scott**
Business Manager: **Ben Moore**

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each. Copyright 1979 by POPULAR COMPUTING.

@ 2023 This work is licensed under CC BY-NC-SA 4.0

Computniks

by A. Cynic

A computnik is a person who "gets his kicks" from numbers and/or from computing itself. Long before modern computers we had computniks around. A good example is Shanks who long ago, using hand methods, tried to compute pi to 707 places (he made a mistake along the way). Even he could not have believed that anyone needed that many digits of pi, or that in the foreseeable future they would be of any use except to set a record. Furthermore, he must have known that he could not thereby prove that pi was or was not an algebraic number. No, he merely did it "because it was there" and "to set a record"--the usual replies that a computnik gives when asked, "Why do you do it?" There seems to be a widespread belief that getting into the Guiness book of records is worth the effort; that it is a worthwhile use of one's necessarily limited time and effort in this life. Computniks seem to believe in this kind of importance.

They also often give the justification for their activities that "it amuses me" and that "it doesn't do any harm." Indeed, as a long time professor, I have often heard the same kind of justification given by professors when asked why they included certain material in their courses; namely, "The students like it." Using that standard, then, a laboratory course for 16-year-olds in sex would prove to be very popular and therefore should be given! That a student likes a course, or that a person "gets his kicks" out of something, is hardly a serious justification for an activity. It does not justify the killing of other persons; why, then, of one's own time and life?

Most of the mental arithmetic freaks in the literature were clearly computniks. From their earliest memories they report that they liked to play with numbers, to compute large numbers. By constant practice they developed great mental skills at doing such things. But their usefulness to society, beyond public entertainment, was slight. Only occasionally and by chance does a computnik's work prove to have lasting value. Occasionally a computnik can be temporarily harnessed to work on something useful to society (such as computing tables that are currently needed, or writing programs that have some immediate value).

That the lure of playing with numbers is widespread and of long standing can be seen from the numerology of the Middle Ages--especially in the numerological efforts to compute the number of the beast (666) from a person's name. Number puzzles in magazines are another proof, and they go back to the earliest Egyptian records of mathematics with their unrealistic problems. Humans have other weaknesses that we do not generally condone, but rather try to alter into more socially useful paths. Pandering to human weaknesses as is so often done, especially by advertisters, is not so innocent an activity that I am willing to automatically condone it.

Modern computer programming provides a new aspect to a computnik's life. The act of programming is a similar monotonous task, similar in the sense that both hand computing and programming require no deep thinking, only coping with a vast sea of essentially trivial details. And it has correspondingly the same lure of activity that avoids deep thinking. How joyous to be active and avoid thinking about any real meaning in life--to be busy with problems that are hardly more complex than jigsaw puzzles. That the program may be based on serious mathematics has little effect on the computnik's pleasure; it is sufficient if it occupies his time and attention.

His "kicks" come from the final running of the program, and he apparently is seldom bothered if he took 99 trials to get it (apparently) running. So long as it is activity, it doesn't matter how wasteful of his or machine time the programming is--of course, for style the final program should have cute time savings that are usually more than offset by the recompile times used to get the program running. It is characteristic of the computnik that he does not want to do any preliminary planning, careful thought, or deep thinking so that the program will run properly the first time. Why take time to plan like an adult when you can play like a child? Besides, playing with the machine is the object, not getting the results.

Computniks are typically parasites on society; they are mainly consumers and seldom creators of wealth. When you consider all the people who must labor to produce the food, shelter, transportation, and education for the individual person so he can even live out his life, then you are brought face to face with the question of how much a person contributes to the society in which he lives. Many people, especially computniks, refuse to answer the question in any serious way. But if there is not a net contribution to the capital of society in each generation, then society will hardly be able to advance. We have already too many parasites on the body of society, people who take much more than they contribute, to encourage people to become computniks, thus adding to the total.

That there is an obligation to contribute to society is illustrated by the following story. The famous number theory expert, L. E. Dickson, once said in class, "When you consider where you would be if previous generations of mathematicians had not labored to organize, prune, simplify, and clearly present mathematical results, then you see that your knowledge of mathematics would be on the level of a caveman's." He went on to say that his contribution, which was a labor of love, was his History of Number Theory.

When you contemplate all that you take from society from birth until you finish school, and then all that you later take throughout your life, there is, I say, a great implied debt. Of course, you need not acknowledge the debt; most people do not.

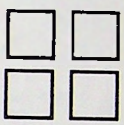
The question of evaluating the life of a computnik and other parasites comes down to a set of values, and all such sets require some basic assumptions. For example, Socrates said, "The unexamined life is not worth living." It is a powerful statement of an attitude toward the meaning of life, but in the final analysis it is merely an opinion and others need not subscribe to it. I have, by repeatedly using the pejorative word "parasite," made the assumption that one has an obligation to use one's abilities in socially useful ways to contribute to society more than one takes away.

In judging the worth of a life spent as a computnik, outsiders can apply their own values, and they can try to force the computnik to face up to his implied set of values when he says that it amuses him and that this is sufficient justification--besides, it does no harm, he says. A famous pure mathematician, G. H. Hardy, in writing his Mathematician's Apology was reduced to saying, "I didn't do any harm." How pathetic was that remark when I first read it as a young aspiring pure mathematician! With but one life to live on this earth, was I to be reduced to saying, when I came to the end, "I didn't do any harm!" How much better to feel that one has contributed at least some small amount to the continual advancement of the human race.

It is hard to know when you have done good or harm; on this point it is easy to be glib and superficial; but the person who examines his life as Socrates advised--indeed, regularly reexamines his life--is forced to apply reasonable prudence in answering the question of his life's contribution to society. To say that one cannot be sure is to say the obvious, and nothing more. To try seriously to assess one's contributions, with decent regard to human fallibilities and self-delusions (and especially the last!) requires the efforts of a mature person and is usually beyond computniks. For them it is sufficient, "I get my kicks from computing."

My Collegiate Dictionary gives:

cynic — One of a Greek school of philosophers who taught that virtue is the only good, that its essence lies in self-control and independence. Later Cynics were violent critics of social customs and current philosophies.



The above anonymous diatribe expresses a point of view. It may be a common point of view, so I concluded, after some soul-searching, that it should be printed. I have written a rebuttal and will present it in the next issue, although a case could be made for believing that the essay is almost self-refuting. The rebuttal is delayed to allow for other responses, which are hereby solicited.

...fg

Primes, Anyone?

We wish to describe in detail how any personal computer can be used to perform some high-precision arithmetic. Specifically, we will show how it is possible to sift out prime numbers in the one billion range.

It is, of course, possible to chain 8-bit words together so as to convert an 8-bit machine into a 16-bit, 32-bit, or almost any-bit processor, functioning directly in binary. This is tricky, however, and best left to the experts. Novices, like me, can get results with much less effort by operating in decimal, using a word of storage to hold one decimal digit. With an 8-bit machine, this is not too wasteful.

The MAIN flowchart is shown as Figure J. For each value of N to be tested for primality, N will be divided by every odd number up to and including the square root of N. (Yes, this is an inefficient algorithm. It has some virtues, though; namely, it is easy to code, it requires little storage, and it is easy to understand.) If any such division yields a remainder of zero, then N is not prime and (at Reference 7) we try a new (odd) value of N. If all the divisions have a non-zero remainder, then N is prime.

To be precise, at Reference 8, we should compare the square of D to N. Since we are proposing high precision work, it is cheaper to set a safe limit for the highest possible D value for all values of N in the range we will be exploring, and this limit is established at Reference 1. For example, for divisors from 3 to 31623, the primes from one billion up to 1,000,014,129 can be sifted.

At References 6 and 7 the same task is performed; namely, to add 2 to a multi-digit number. Flowchart K shows the logic of that operation.

With one digit per word, even such a simple task as testing R for zero (Reference 4) requires a flowchart of its own. We show it as a subroutine in Figure L.

If the value of the divisor, D, is outputted at the point marked (*), together with N, then the program will find the smallest prime factor of a composite number as well as a table of primes.

While everything in this article can be taken in general, the calculation for prime numbers will be described in terms of a specific implementation, to fit most personal computers.

N (the number under test, initialized to an odd starting value) is in words addressed at 1401 through 140E; see Figure P for a storage layout. Addresses for our 8-bit machine are in hexadecimal. Field N is thus 14 decimal digits long, allowing for sifting out primes in the range around one billion.

Each digit of N is addressed as Y = 1401,X; that is, address 1401 plus the contents of the X index register. For left-to-right operations, XR (the index register) is initialized to zero and tested out at 0D; for right-to-left operations, XR is initialized to 0D and tested out at zero. R is a working area in which the divisions on N are performed; R occupies words 1421 through 142E. Each digit of R is addressed as 1421,X, where the XR ranges from zero to 0D.

The nub of the problem, of course, is the long division called for by N/D. The logic is outlined in flowcharts M1 and M2. The division is done much as it is done on paper, except that it is easier to add the complement of D than to subtract D. This requires another subroutine--to form the complement of DS and put it at DC--whose logic is shown in Figure N.

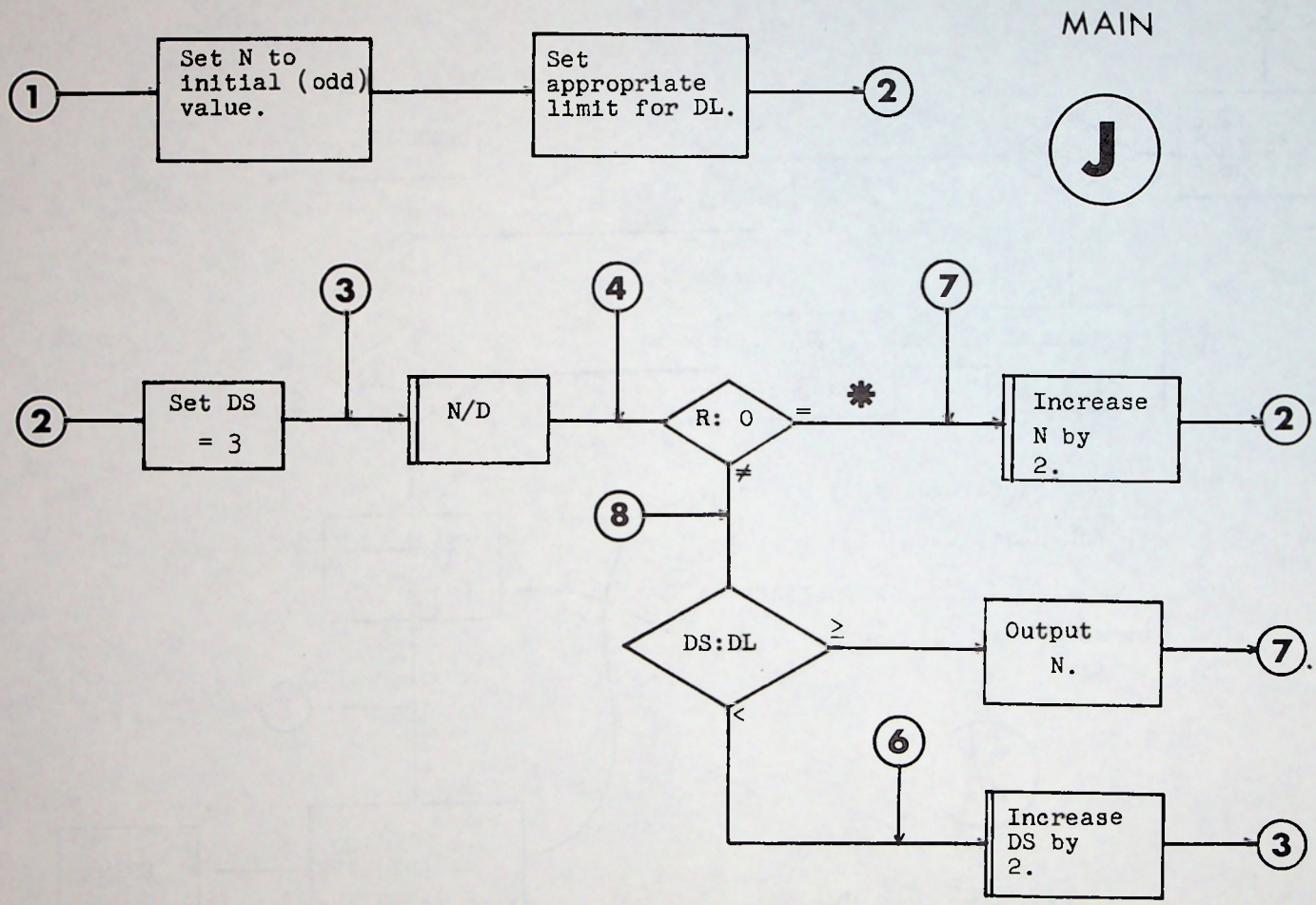
The logic of the division operation is as follows. We begin by lining up the first six digits of R with DA:

1421 1422 1423 1424 1425 1426

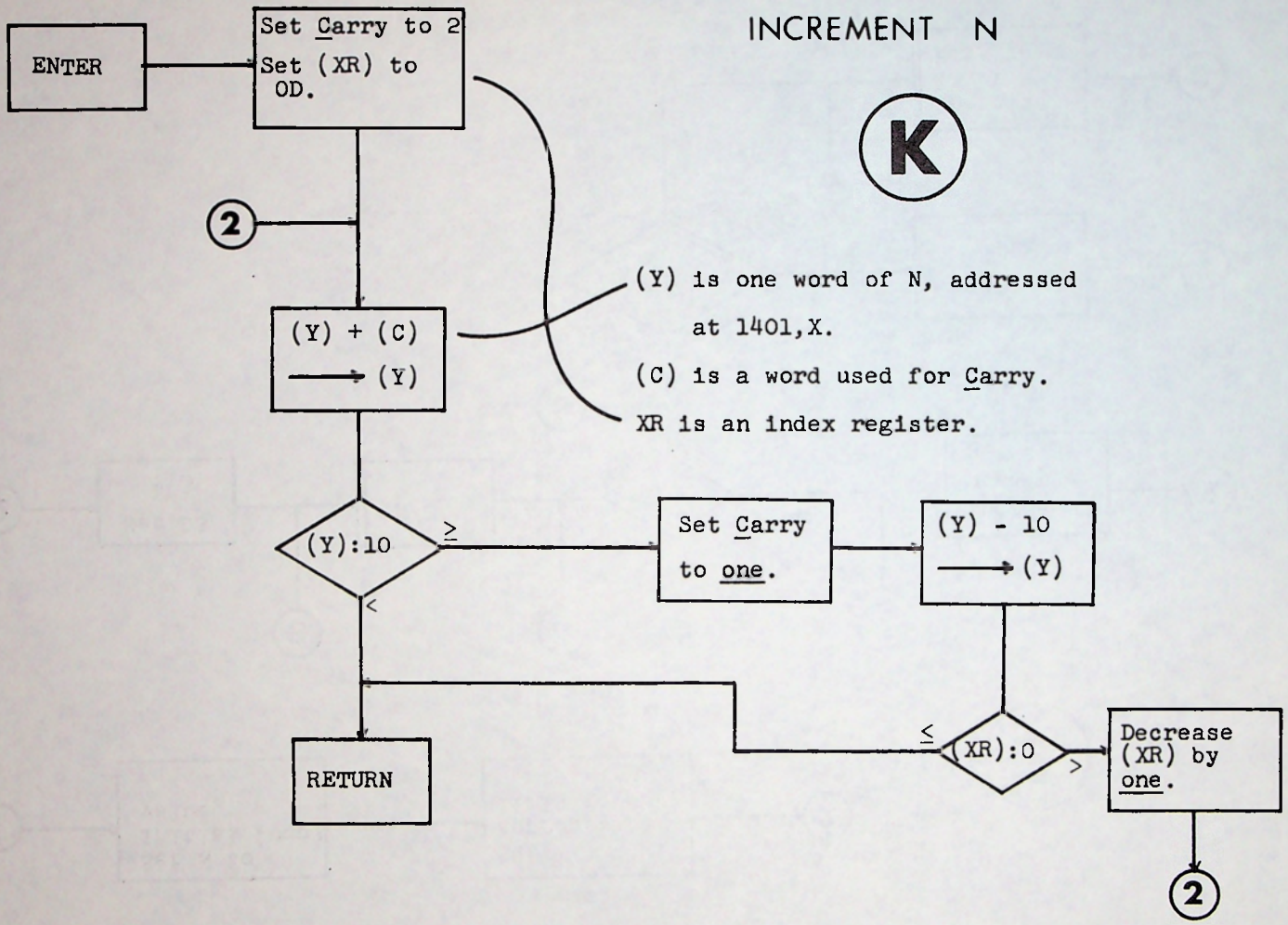
1501 1502 1503 1504 1505 1506

and we subtract (that is, add the ten's complement) repeatedly, checking (at Reference 6) for going negative, which will be signaled by the presence of a 9 in the first position of the six working positions of R. We are thus simulating the operation of a rotary mechanical calculator. When the result of the subtractions goes negative, we add for one cycle. In order to use the same instructions for both the additions and subtractions, the switch at Reference 4 controls the action. The notation R at Reference 3 refers to one digit (that is, the contents of one word) of field R. At references 6 and 10, the R refers to the address of the first word of the six words involved in the current subtraction or addition. In threading through the logic, it must always be kept in mind that we are performing decimal arithmetic, one digit per word.

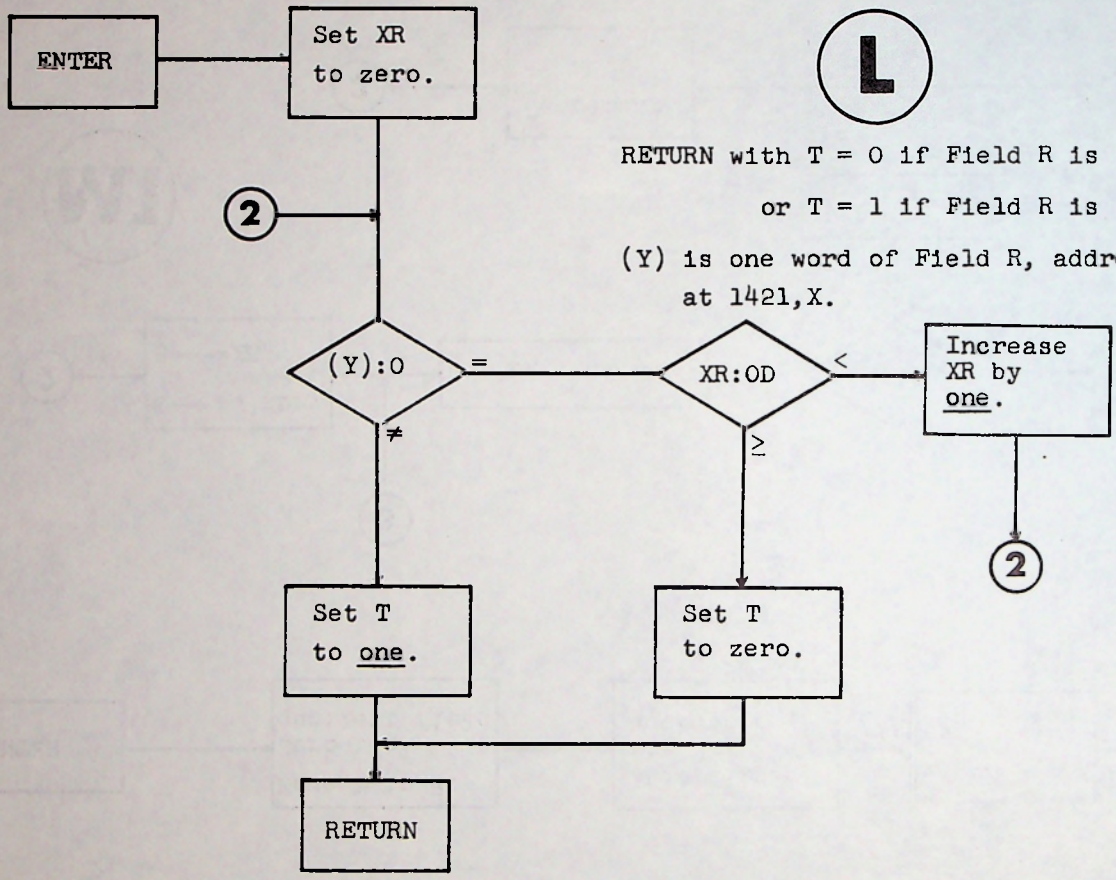
The table that accompanies this article provides test data for the next implementation. Additions to this table are solicited. Of greater interest, however, is a revised version of this program that will handle numbers in higher ranges.


MAIN
J
1
Set N to initial (odd) value.
Set appropriate limit for DL.
2
2
Set DS = 3
3
N/D
4
R: 0
=
*
7
Increase N by 2.
2
≠
8
DS:DL
≥
Output N.
7
<
6
Increase DS by 2.
3

INCREMENT N


ENTER
Set Carry to 2
Set (XR) to OD.
2
(Y) + (C) → (Y)
(Y) is one word of N, addressed at 1401,X.
(C) is a word used for Carry.
XR is an index register.
(Y):10
≥
Set Carry to one.
(Y) - 10 → (Y)
<
RETURN
≤
(XR):0
>
Decrease (XR) by one.
2

TEST R FOR ZERO



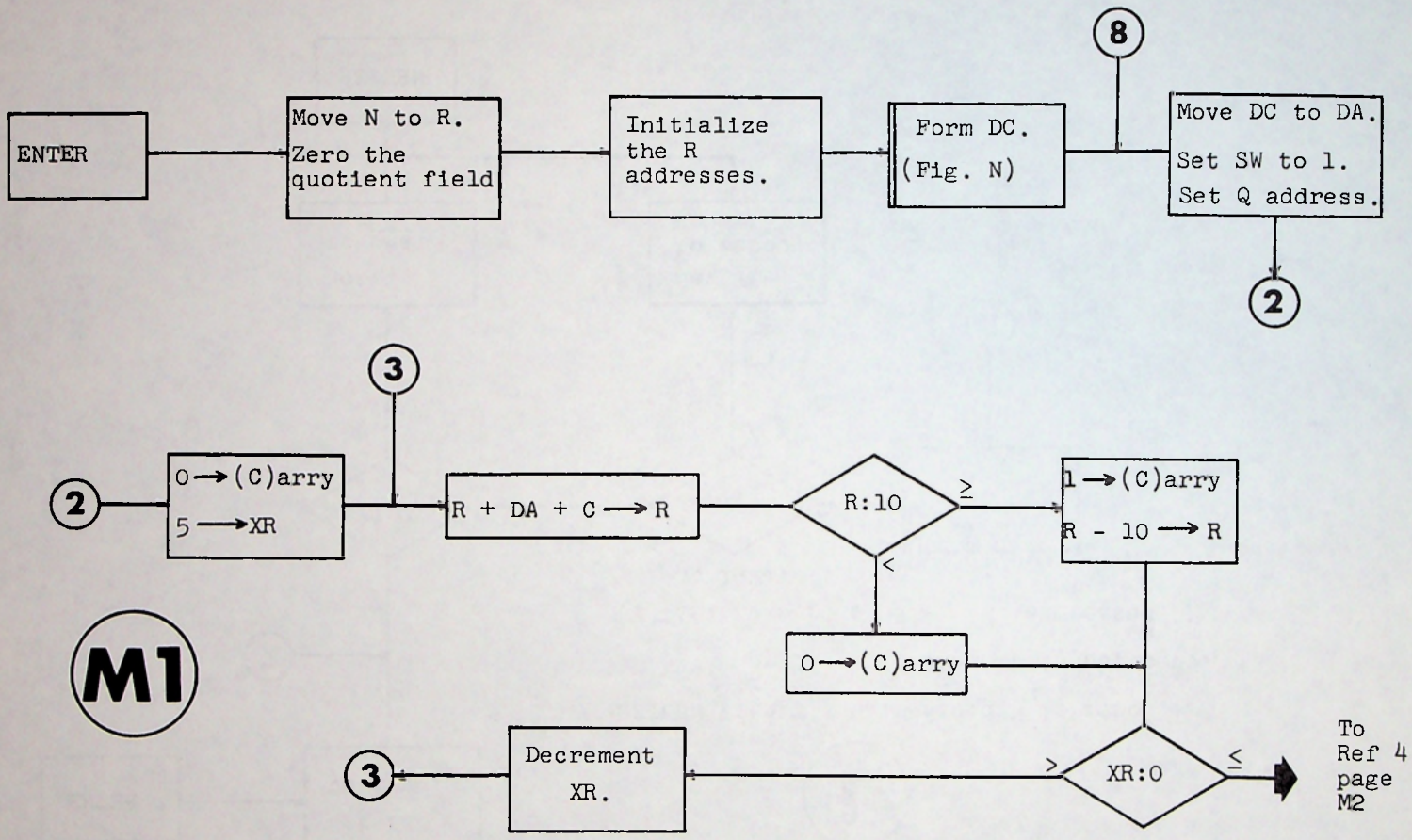
L

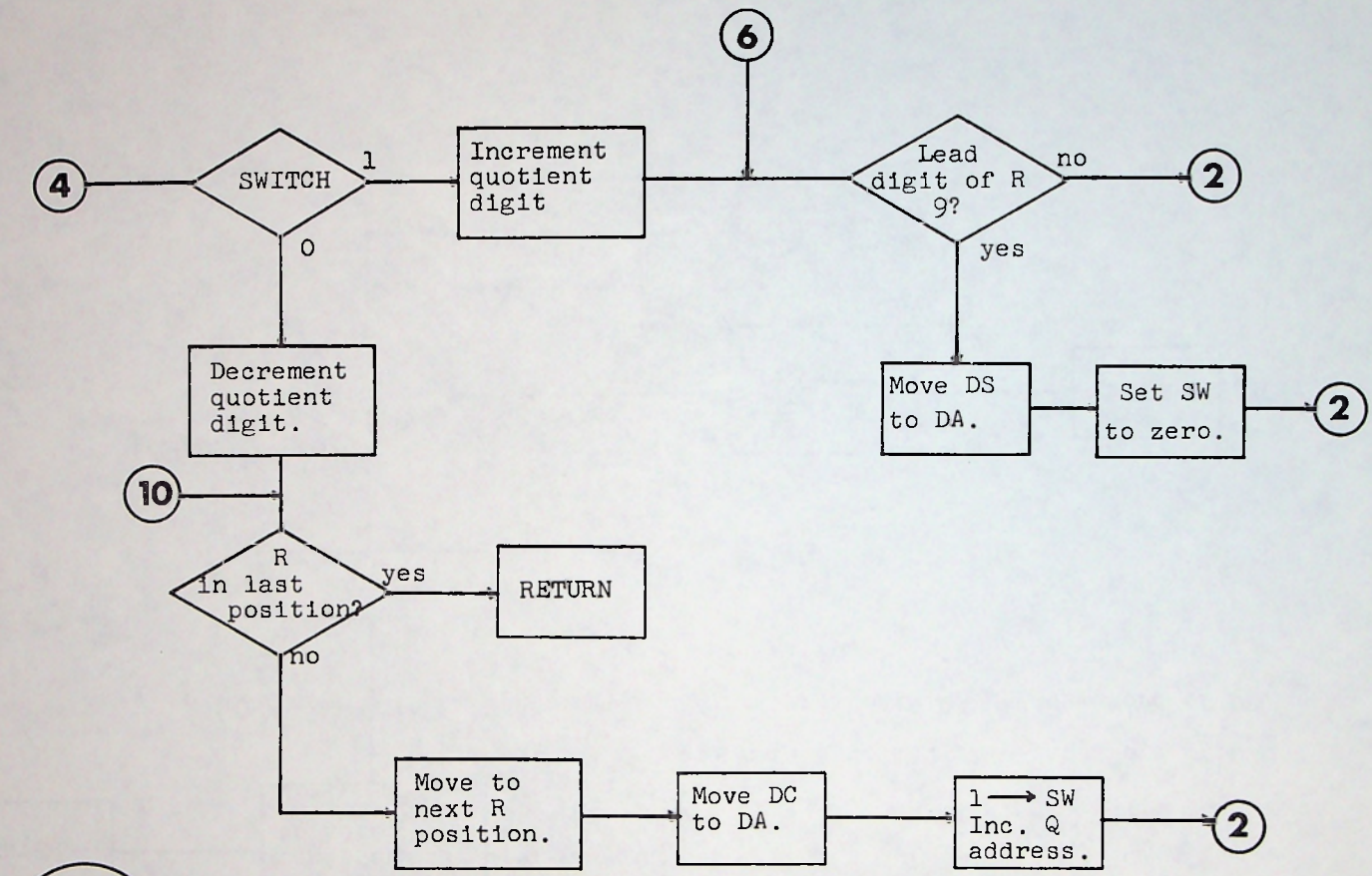
RETURN with T = 0 if Field R is zero.

or T = 1 if Field R is not zero.

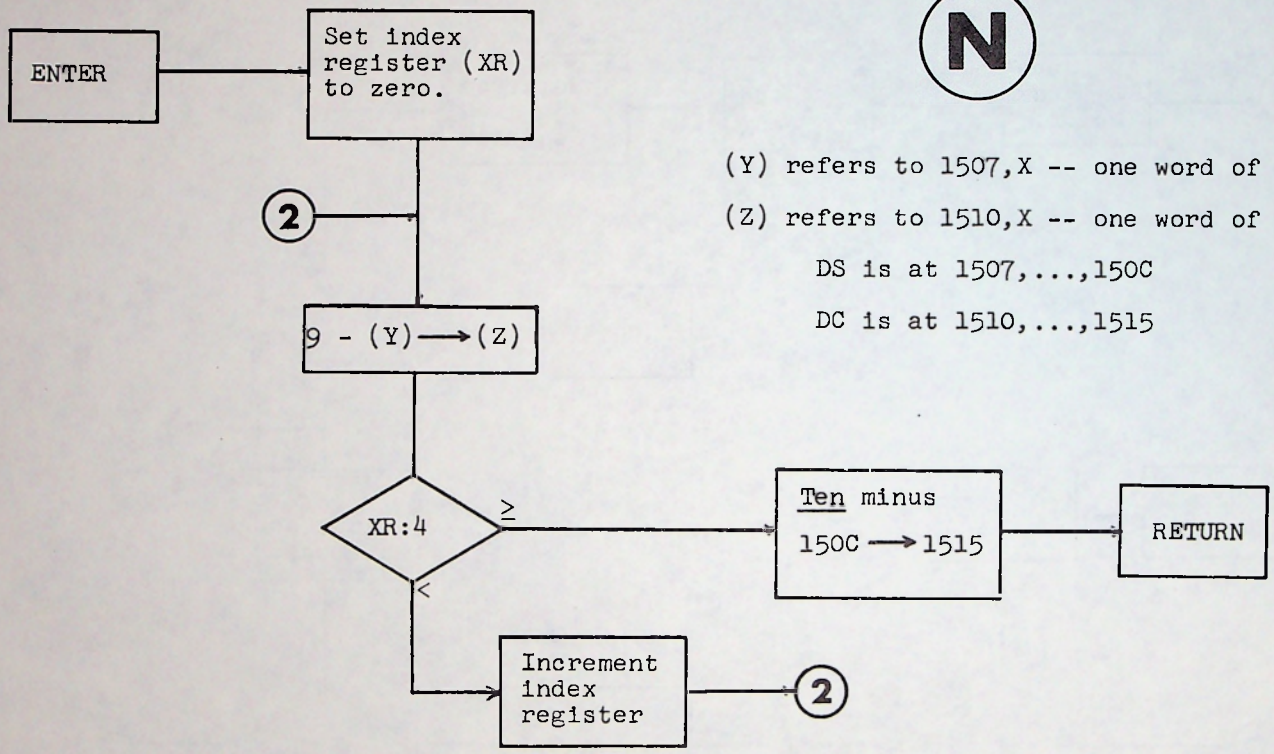
(Y) is one word of Field R, addressed at 1421,X.

N/DA


ENTER
Move N to R.
Zero the quotient field
Initialize the R addresses.
Form DC.
(Fig. N)
8
Move DC to DA.
Set SW to 1.
Set Q address.
2
3
2
0 → (C)arry
5 → XR
R + DA + C → R
R:10
≥
1 → (C)arry
R - 10 → R
<
0 → (C)arry
M1
3
Decrement XR.
>
XR:0
≤
To Ref 4 page M2


6
4
SWITCH
1
Increment quotient digit
Lead digit of R 9?
no
2
0
yes
Decrement quotient digit.
Move DS to DA.
Set SW to zero.
2
10
R in last position?
yes
RETURN
no
Move to next R position.
Move DC to DA.
1 → SW Inc. Q address.
2
M2

FORM COMPLEMENT OF DS



N

(Y) refers to 1507,X -- one word of DS.

(Z) refers to 1510,X -- one word of DC.

DS is at 1507,...,150C

DC is at 1510,...,1515

1401	1402	1403	1404	1405	1406	1407	1408	1409	140A	140B	140C	140D	140E **N**
1421	1422	1423	1424	1425	1426	1427	1428	1429	142A	142B	142C	142D	142E **R**

1501	1502	1503	1504	1505	1506	**DA**	The Active Divisor
1507	1508	1509	150A	150B	150C	**DS**	The Divisor Source
1510	1511	1512	1513	1514	1515	**DC**	Complement of DS
1516	1517	1518	1519	151A	151B	**D3**	Constant, three
1520	1521	1522	1523	1524	1525	**D3C**	3 Complement, 999997
1526	1527	1528	1529	152A	152B	**DL**	Limit on D

P

PC80-16

1 000 000 000+

0007	009	021	033	087	093	097	103	123	181	207	223
0241	271	289	297	321	349	363	403	409	411	427	433
0439	447	453	459	483	513	531	579	607	613	637	663
0711	753	787	801	829	861	871	891	901	919	931	933
0993	011	021	053	087	099	137	161	203	213	237	263
1269	273	279	311	329	333	351	371	393	413	447	449
1491	501	531	537	539	581	617	621	633	647	663	677
1699	759	773	789	791	801	803	819	857	887	917	927
1957	963	969	043	089	103	139	149	161	173	187	193
2233	239	277	307	359	361	431	449	457	499	571	581
2607	631	637	649	667	727	791	803	821	823	827	907
2937	989	009	013	051	057	097	111	133	153	157	163
3211	241	247	253	267	271	273	283	309	337	351	357
3373	379	397	469	471	513	519	559	577	579	601	621
3643	651	663	679	709	747	751	769	777	787	793	843
3853	871	889	891	909	919	931	951	957	967	987	999
4023	059	099	119	123	207	233	249	251	263	321	329
4381	389	437	449	459	497	507	519	539	567	569	581
4609	611	627	633	647	693	699	717	771	777	783	791
4807	839	843	849	857	867	869	891	893	897	927	933
4977	981	001	029	053	067	103	133	187	197	203	233
5277	287	299	317	329	341	353	361	403	407	437	443
5449	451	469	491	527	541	547	583	631	647	683	731
5737	751	763	791	823	827	833	847	863	899	907	947
5953	959	971	973	991	997	019	027	037	039	061	093
6099	127	129	177	193	211	223	229	307	313	331	349
6379	393	417	421	457	459	477	541	571	577	583	607
6621	661	663	697	717	751	781	837	867	901	957	961
6967	981	003	023	027	089	117	137	147	159	191	209
7237	243	257	279	321	347	383	389	417	429	447	453
7467	479	497	513	521	531	537	557	633	647	651	653
7681	707	719	741	759	773	803	819	837	873	887	909
7923	927	929	941	041	083	089	101	109	127	173	181
8223	257	259	271	277	311	313	343	349	397	403	419
8431	439	487	511	557	593	617	637	649	661	671	679
8719	727	761	773	791	797	803	811	829	853	899	917
8937	967	009	013	063	081	093	099	123	133	163	183
9211	223	259	277	279	289	301	321	331	363	399	403
9421	441	457	469	487	519	529	531	541	559	561	567
9573	579	597	601	609	631	651	667	679	711	733	739
9757	789	831	859	867	961	999	029	051	069	101	153
10173	189	197	233	243	251	267	281	303	321	327	351
10357	381	449	467	483	503	513	549	593	597	611	633
10659	699	707	723	747	749	761	773	777	801	833	903
10953	969	971	981	987	007	011	071	083	091	107	109
11137	149	161	223	239	253	269	277	283	289	301	317
11329	371	377	391	421	479	487	497	517	533	539	559
11583	601	619	631	659	673	679	707	763	767	769	773
11799	811	821	823	847	869	967	983	989	019	037	121
12157	177	187	217	219	231	241	253	297	309	333	337

Commentary

...from James L. Boettler, Mathematics Department of Claflin College, Orangeburg, South Carolina:

Many computer books include flowcharts of such non-computer things as changing a flat tire, getting out of bed in the morning, getting ready to leave for work, etc. In just about every case, the author(s) present a humorous solution.

Sure, it's good to see some humor in our texts. But when the only humor that appears is in these flowcharts, students are not likely to take other flowcharts seriously.

I'm becoming convinced that most students have more trouble figuring out the logic of a problem solution than in coding the solution in a computer language. So, this term, I spent far more time in teaching algorithms than in computer languages. The text I used was certainly conducive to this approach: Olivieri and Rubin's Computers and Programming: A Neoclassical Approach.

I went beyond the text, and made a list of problems which nearly every student has solved sometime in the past. Some of these are:

1. Put on a T-shirt that is inside out.
2. Get channel 6 working well on TV.
3. Get your car started.
4. Find "Bob Jones" in the telephone directory, with address and phone number.
5. Find the definition of "program" in your dictionary.
6. Find verse 21 chapter 5 of I Thessalonians in the Bible (Note: this verse is good advice for programmers).
7. Tie your shoes.
8. Take a shower.
9. Walk from Claflin College to Edisto Gardens.
10. Register for your college courses.
11. Parallel-park your car between two other cars.
12. Cross Highway 301 (4 lanes with no dividers)-- on foot.
13. Cross Interstate 26 (4 lanes with divider)-- on foot.

(In these problems, include the most critical steps at the main levels.)

Mr. Boettler continued:

I assigned these problems to my students to analyze. They were to present a solution to each problem either using flowcharts or pseudo-code. Most of the problems were rather open-ended; one could write a whole book on how to walk. So I told my students they could limit themselves to half to a whole page. I also asked them to include a list of the assumptions they made in solving their problems.

The results were not very encouraging. For example, in the problem of how to take a shower, only one student bothered to turn the water off after he was done. But yet, I feel that the students can gain a lot of insight into programming by working on such problems. They should appreciate that English is not precise enough and see why we create special computer languages for programming.

Correction

Professor Gary Wicklund, of the University of Iowa College of Business Administration, points out an error in the BASIC program (issue 77, page 4) to calculate great circle distances.

"We discovered a difficulty with line 240 of the program. The arctangent (ATN) function gives values for the angle in radians from $-\pi/2$ to $\pi/2$; therefore, if the angle is less than zero a negative distance is calculated; for example:

Tokyo to Lima, where X = -0.7560

$$a = \tan^{-1}\left(\frac{\sqrt{1 - (-.7560)^2}}{-.7560}\right) = -40.8872 \text{ degrees}$$

$$= -.71360 \text{ radians}$$

$$\alpha = \cos^{-1}(-.7560) = 139.1128 \text{ degrees}$$

$$= 2.42798 \text{ radians}$$

To correct this problem, I added two statements as follows:

```
242 IF (A>=0) THEN 250
244 A = A + 3.14159265
```

Further research on the 3X+1 problem (see issue 79 for details about the problem) has been done by Contributing Editor Ed Ryan. Using a program that could handle numbers up to 250 decimal digits long, the 3X+1 algorithm was executed on N values up to 240 digits.

A table of the A values at the 35-digit level is given.

Average A values over the range of N from one digit to 240 digits were calculated, and a least squares curve fit performed on that data, with this result:

$$A = 24.64D - 101.74$$

where D is the number of digits and A is the expected value for A. For example, for 100-digit values of N, the expected A values will be around 2363 by the above formula. (Observed values of A at 100-digit values of N average 2183.) The data for the prediction formula was calculated by John Charles Ryan.

The largest A value so far observed is 6451, for N =

100,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,099,900,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
009,990,000,000,000,000,999,000,000,000,000,099,909,990,000,
000,000,000,000,027

A values for N =

10,000,000,000,000,000,000,000,000,000,000,000

and the following 239 consecutive integers:

551	657	657	657	657	657	657	657	657	657
657	657	657	657	657	657	551	657	657	657
551	551	657	657	551	657	551	975	657	657
657	975	551	657	551	551	657	657	657	657
551	869	551	657	551	551	551	869	551	657
657	657	551	551	869	869	551	657	551	657
551	551	869	869	551	657	657	657	551	551
551	975	551	869	551	551	551	551	657	657
551	551	869	869	551	551	657	657	551	657
551	869	551	551	869	869	551	869	551	551
551	551	551	657	657	657	551	657	962	962
962	657	551	551	657	657	551	551	657	657
551	869	551	657	962	962	962	657	551	962
551	551	551	551	551	657	551	869	551	657
551	551	657	657	551	657	657	657	551	551
551	551	551	657	551	657	657	657	657	975
551	975	551	551	869	869	869	869	551	657
551	962	551	551	551	962	551	551	551	551
551	551	962	962	551	657	551	657	869	869
869	657	551	657	657	657	551	551	551	657
551	551	551	657	551	551	657	657	551	657
657	657	551	551	551	869	869	551	869	657
869	869	869	869	551	657	551	551	551	551
551	962	551	869	551	551	869	551	551	869